Assignment 4: Content Based Image Retrieval
Vision and Image Processing
Hold 2 - Assignment group 5

Marcus Frehr Friis-Hansen: lns611
Mikkel Keller Andreasen: trq281
Søren Strøyberg: lds147

December 2025

Comments on codebook generation and indexing

We took inspiration from the TA Session 4 slides "VIP Exercise 4", n.d. on the method of which we used to import data. As suggested we loaded the data using **HuggingFace**. This saved us the effort of manually downloading the dataset. Also instead of manually selecting which labels we would use in our pipeline, we simply randomly selected a given number of labels, using a random seed for reproducibility. We also extracted the SIFT features as suggested, using OpenCV's implementation. For the rest of the code-book generation, we simply did as described in the assignment, using scikit-learns kmeans-clustering implementation for clustering, and pandas to create the table where each row contains a filename, label, which split the file belongs to and the corresponding Bag of Words (BOW).

Argue for choice of retrieval method

Common words is the simplest retrieval method. It works by simply comparing how many visual words does the query image have in common with the images in the table described earlier, and ranking them in descending order. While we do not expect the method to perform well, we will be using it as a basic performance measure. How well does the other retrieval methods perform when compared to Common Words.

Term Frequency- Inverse Document Frequency is a more complex retrieval method. It works by giving each visual word a weight in relation to how descriptive or important the word is. The method builds upon the idea that certain words, although common, aren't that descriptive. Words like 'and' , 'but', or 'or' (conjunctions etc.). IDF works by down-weighting words that appear frequently over multiple images, such that rarer and more descriptive visual words gets higher importance. For a given word, this is done like shown below

$$idf_w = \frac{\text{Number of documents or images}}{\text{number of documents in which w appears}}$$

Yin, n.d.

Essentially tf-idf emphasizes words that are frequent within the query image, but rare across the dataset. We used scikit-learn implementation **TfidfTransformer** which works by creating an object that can both learn the weights using the bow-matrix and then transform given bow-histograms. It also uses euclidean normalization (L2) to make sure that some rare words don't out scale the other visual words. Finally when the query image is compared to the rest of the dataset, they are compared using **Cosine_similarity** as recommended in the slides. As smaller angle describes images that are similar. Also note that since we have L2-normalized, the cosine similarity is equal to the dot product

$$\frac{v_d \cdot v_q}{||v_d||||v_q||} = v_d \cdot v_q, \quad \text{when} \quad ||v_d||||v_q|| = 1$$

Although not completely necessary, we decided to keep the **Cosine_similarity** to compare BOW in our code, as it was simpler to implement than using the dot-product. We expect that tf-idf will perform better than the other retrieval methods.

Bhattacharyya distance compares vectors using the formular shown below. It is meant to be used on normalized vectors, which we took into account during implementation. Bow-histograms that contain similar words will also get a higher similarity score, and will therefore rank higher. We expect that the retrieval method which uses this distance measure will perform better than common-words, but similar to Kullback-Leibler divergence.

$$d(v_1, v_2) = \sum_{i=1}^{k} \sqrt{v_{1i} v_{2i}}$$

Yin, n.d.

Kullback-Leibler divergence is used to quantify how dissimilar images is to on another, and is meant to be used on normalized vectors. Images that are similar to the query image will have a lower score, and should therefore be ranked higher than images with a high score. That is also why we sort in ascending order in our implementation. We expect that that this distance measure will perform better than common words, and similar to Bhattacharyya distance.

$$D_{KL}(v_1|v_2) = \sum_{i=1}^{K} v_{1i} \ln \frac{v_{1i}}{v_{2i}}$$

Yin, n.d.

Overall, due to tf-idf's more sophisticated comparison method, we believe it will perform better than the other methods, in the two experiments described below. We expect that the relative ranking between the methods will look something like. 1. tf-idf, 2. Bhattacharyya and Kullback-Leibler, 3. Common words

Experiments

In the first experiment below, we test how well we can retrieve the training images, and in the second we test how well we can classify the test images, both using a query image. The two experiments are identical, expect for the data used for the experiment, training data and test data respectively. Both experiments use the metrics:

Mean reciprocal rank, which is the average rank across all queries.

$$\text{mean reciprocal rank} = \frac{\sum_{i=1}^{N} \frac{1}{\text{rank}_i}}{N}$$

How often in percent the correct category is in the top-3 across all queries. i.e. how often the correct category is in the top 3 on average. Also note that for these experiments we used 20 random labels, and a set random seed for reproducibility. We also tested different random seeds to see how different labels impacted performance metrics.

Experiment 1 - training data

Random Seed	Method	Reciprocal Rank	Top-3 Accuracy
39	Common Words	0.2527	0.2797
39	TF-IDF	0.6277	0.6935
39	Bhattacharyya	0.5687	0.6524
39	Kullback–Leibler	0.3756	0.5109
37	Common Words	0.2014	0.1854
37	TF-IDF	0.5711	0.6160
37	Bhattacharyya	0.3585	0.3660
37	Kullback–Leibler	0.3818	0.5654
36	Common Words	0.3563	0.3585
36	TF-IDF	0.5529	0.5794
36	Bhattacharyya	0.5020	0.5146
36	Kullback–Leibler	0.4027	0.4114
Mean	Common Words	0.2701	0.2745
Mean	TF-IDF	0.5839	0.6296
Mean	Bhattacharyya	0.4764	0.5110
Mean	Kullback–Leibler	0.3867	0.4959

Table 1: Retrieval performance for Experiment 1 (training data) across different random seeds and retrieval methods. The final rows indicate mean performance across seeds

as expected common words performed relatively poorly with an average of 0.2701 reciprocal rank, and 0.2745 top 3 accuracy across randoms seeds. For the same metrics, Kullback-Leibler performed better at 0.3867 and 0.4959, followed by Bhattacharyya at 0.4764 and 0.5110. As expected this makes tf-idf the best performing retrieval method at 0.5839 and 0.6296 for reciprocal rank and top-3 accuracy respectively, on average across random seeds for the training set.

Experiment 2 - test data

Random Seed	Method	Reciprocal Rank	Top-3 Accuracy
39	Common Words	0.2585	0.2889
39	TF-IDF	0.6358	0.6943
39	Bhattacharyya	0.5794	0.6566
39	Kullback–Leibler	0.3872	0.5226
37	Common Words	0.1993	0.1869
37	TF-IDF	0.5651	0.5958
37	Bhattacharyya	0.3506	0.3450
37	Kullback–Leibler	0.3724	0.5491
36	Common Words	0.3142	0.3069
36	TF-IDF	0.5497	0.5926
36	Bhattacharyya	0.4879	0.5185
36	Kullback–Leibler	0.3606	0.3558
Mean	Common Words	0.2573	0.2609
Mean	TF-IDF	0.5835	0.6276
Mean	Bhattacharyya	0.4726	0.5067
Mean	Kullback–Leibler	0.3734	0.4758

Table 2: Retrieval performance for Experiment 2 (test data) across different random seeds and retrieval methods. The final rows indicate mean performance across seeds.

For the test-set the same trend appears with common words at 0.2573 and 0.2609, Kullback-Leibler at

0.3734 and 0.4758, Bhattacharyya at 0.4726 and 0.5067, and tf-idf at 0.5835 and 0.6276. All these metrics mentioned are of course, reciprocal rank and top-3 accuracy, on average across queries and randoms seeds.

Discussion

In both table 1 and table 2 above, similar results can be seen for both experiments on average across seeds. This shows that the implemented Content Based Image Retrieval (CBIR) implementation is robust and generalizes well to unseen data i.e the test set.

However it is interesting how much the performance of each retrieval method varies based on the random seed. F.X. in experiment 2 Bhattacharyya retrieval varies between $|0.5794 - 0.3506| = 0.2288$ for reciprocal rank, and $|0.6566 - 0.3450| = 0.3116$ for top-3 accuracy, between seed 39 and seed 37. We believe this effect is caused by the different labels used for each seed. Labels that are visually similar might cause the performance to drop. E.g. the labels "sunflower" and "pizza" could both be described using a lot of the same visual words.

This phenomenon is much less apparent for the tf-idf retrieval method as it only varies between $|0.6358 - 0.5497| = 0.0861$ and $|0.6943 - 0.5926| = 0.1017$, which suggests that the method is more stable against different features across similar categories.

Conclusion

Overall as tf-idf showed higher accuracy across both experiments and different seeds, while also being less prone to variation in accuracy, we would argue it is superior to the other retrieval methods, and would therefor use tf-idf if we were to implement a CBIR system in the future. The other methods can still be used to compare if needed. We would however refrain form using Bhattacharyya similarity as its accuracy varies to much, making it unreliable for comparison.

References

Vip exercise 4 [slides]. (n.d.).
Yin, S. I. O. H. (n.d.). *Content based image retrieval* [slides].